AFHRL-TP-90-53

AIR FORCE

AD-A224 758

HUMAN RESOURCES

LABORATORY

OBJECT-ORIENTED INTEGRATED MAINTENANCE INFORMATION SYSTEM (IMIS) DIAGNOSTIC MODULE DEVELOPMENT

Garth Cooke
Johnnie Jernigan
Nicola Maiorana
Theodore Myers

Systems Exploration, Incorporated
5200 Springfield Pike, Suite 312
Dayton, Ohio 45431

Janet E. Murphy, 2Lt, USAF
Eric N. Carlson, 2Lt, USAF

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

July 1990

Final Technical Paper for Period September 1989 - June 1990

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

JANET E. MURPHY, 2Lt, USAF
Task Monitor

BERTRAM W. CREAM, Technical Director
Logistics and Human Factors Division

JAMES C. CLARK, Colonel, USAF
Chief, Logistics and Human Factors Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1990	Final Paper - September 1989 - June 1990

4. TITLE AND SUBTITLE
Object-Oriented Integrated Maintenance Information System (IMIS)
Diagnostic Module Development

5. FUNDING NUMBERS
C - F33615-88-C-0004
PE - 62205F
PR - 1710
TA - 00
WU - 40

6. AUTHOR(S)
Garth Cooke
Johnnie Jernigan
Nicola Maiorana
Theodore Myers
Janet E. Murphy
Eric N. Carlson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Systems Exploration, Incorporated
5200 Springfield Pike, Suite 312
Dayton, Ohio 45431

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
Logistics and Human Factors Division
Air Force Human Resources Laboratory
Wright-Patterson Air Force Base, Ohio 45433-6503

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFHRL-TP-90-53

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This technical paper summarizes the work performed to redesign and enhance the Integrated Maintenance Information System Diagnostic Module (IMIS-DM). The diagnostic module provides technical support to the maintenance technician by furnishing a wide range of capabilities assisting in the selection of an efficient sequence of maintenance tasks.

One of the major outcomes of this effort was the redesign of the IMIS-DM using Object-Oriented (OO) rapid prototyping techniques to create a diagnostic module compatible with hierarchical data base concepts employed by the AFHRL Content Data Model (CDM). Another major outcome was the enhancement of the IMIS-DM assessment operations. The enhanced IMIS-DM now provides three modes of assessment (functional, physical, and degraded) whereas the previous version provided only the functional mode.

14. SUBJECT TERMS
criticality
diagnostics
fault
functional assessment
physical assessment
rectification
symptom

15. NUMBER OF PAGES
36

16. PRICE CODE

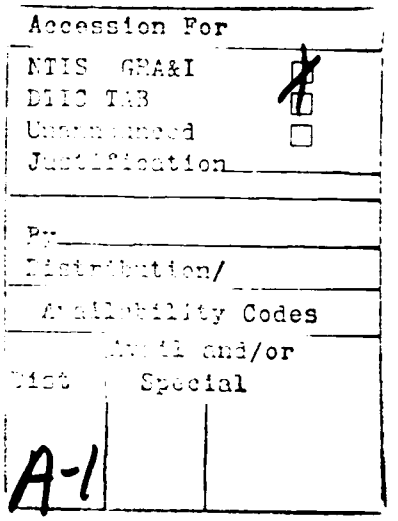
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFHRL Technical Paper 90-53 July 1990

OBJECT - ORIENTED

INTEGRATED MAINTENANCE INFORMATION SYSTEM (IMIS)

DIAGNOSTIC MODULE DEVELOPMENT



Garth Cooke
Johnnie Jernigan
Nicola Maiorana
Theodore Myers

Systems Exploration, Incorporated
5200 Springfield Pike, Suite 312
Dayton, Ohio 45431

Janet E. Murphy, 2Lt, USAF
Eric N. Carlson, 2Lt, USAF

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson AFB, Ohio 45433-6503

Reviewed by

Robert C. Johnson
Chief, Combat Logistics Branch

Submitted for publication by

Bertram W. Cream, Technical Director
Logistics and Human Factors Division

SUMMARY

This technical paper summarizes the work performed by Systems Exploration, Inc. (SEI) to redesign and enhance the Integrated Maintenance Information System Diagnostic Module (IMIS-DM) and diagnostic maintenance environment. The diagnostic module is part of an ongoing IMIS research and development (R&D) effort by the Air Force Human Resources Laboratory (AFHRL) to access and integrate maintenance information from multiple sources and present the information to technicians through a rugged, hand-held computer. The diagnostic module, redesigned in Smalltalk/V and utilizing information from the Content Data Model (CDM), provides technical support to the maintenance technician by furnishing a wide range of capabilities assisting in the selection of an efficient sequence of maintenance tasks.

One of the major outcomes of this effort was the IMIS-DM redesign using Object-Oriented (OO) rapid prototyping techniques to create a diagnostic module compatible with hierarchical data base concepts employed by the AFHRL CDM. Another major outcome was the enhancement of the IMIS-DM assessment operations. The enhanced IMIS-DM now provides three modes of assessment (functional, physical, and degraded) whereas the previous version provided only the functional mode. Other enhancements to the IMIS-DM further expanded diagnostic capabilities with the implementation of algorithms to provide an access group option, "But Not" data entry, presentation of test results, and a revised criticality function.

Enhancements to the IMIS diagnostic maintenance environment provide maintenance technicians with a test result data validity check and the ability to specify unsuccessful maintenance actions. All enhancements, with the exception of the physical assessment module, were implemented in the OO rapid prototyping environment offered by Smalltalk/V. The physical assessment module was implemented in Smalltalk/V logic-based PROLOG.

PREFACE

This paper provides a summary of the Integrated Maintenance Information System Diagnostic Module (IMIS-DM) redesign and enhancement for the Air Force Human Resources Laboratory, Combat Logistics Branch (LRC), under the terms of contract #F33615-88-C-0004, Task Order #0012. The task monitor was Lt. Janet Murphy, AFHRL/LRC.

Research was performed by the Dayton regional office of Systems Exploration, Inc. (SEI). Principal investigators were Garth Cooke, Nicola Maiorana, Theodore Myers, and Johnnie Jernigan.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Purpose

The Air Force Human Resources Laboratory (AFHRL) is engaged in a long-term program to improve information presentation in the maintenance environment. Research and Development (R&D) in the maintenance environment has led to an Integrated Maintenance Information System Diagnostic Module (IMIS-DM) capable of utilizing existing data parameters and producing effective isolation and repair recommendations. This report describes the design of the Object-Oriented (OO) Smalltalk/V IMIS-DM and enhancements. The redesigned diagnostic module was developed using rapid prototyping techniques and the logic-based programming of Smalltalk/V.

Background

The IMIS-DM, or Maintenance Diagnostic Aiding System (MDAS), was a product of earlier R&D efforts and was designed to assist aircraft maintenance technicians in the identification and repair of malfunctioning equipment. A key feature of the diagnostic module is that its algorithms are designed to minimize time to repair an aircraft rather than time to isolate a fault. This philosophy incorporates rectification actions into the overall diagnostic sequence when appropriate, resulting in the repair of faulty components and physical damage in minimum time. The diagnostic module employs special subroutines modifying the split-half dependency model by applying fault/symptom/component matching, component histories, probabilistic data, logistics constraints, and operational constraints.

Fault/symptom matching is employed throughout the diagnostics modeling scheme. The term "fault" is used to describe a functional or physical manifestation of some low level physical failure. Under normal circumstances, it is likely there would be only one fault present in a system at any time. The purpose of a diagnostics task would be to isolate that fault and rectify conditions causing it. Throughout this report, we use the term fault to refer to both the single fault present in the system (that which is bad) and to all possible faults that can cause a symptom. A symptom is a machine-generated code or a verbal description indicating a malfunction exists within a system. The symptom implicates one or more possible faults causing the malfunction. Use of these terms can be confusing in a hierarchically-arranged data base such as the Content Data Model (CDM) because the thing called a fault at one level of the hierarchy may be referred to as a symptom in a lower level of the hierarchy. This is illustrated in Table 1.

In the radar system example shown in Table 1, the lists of possible faults are neither all inclusive nor necessarily correctly stated.The lists are merely illustrative. As each fault is isolated at any level of the hierarchy, that fault can become a descriptive symptom of lower level faults farther down in the hierarchy.

Previous R&D efforts in diagnostics produced the IMIS-DM, an almost purely functional assessment module for isolating and repairing faulty components. Therefore, the software once called MDAS is now designated as the functional assessment module. Below is a brief discussion of the functional assessment module's initialization and operations processes. Detailed information about algorithms and operations can be found in the technical paper, Integrated Maintenance Information System (IMIS) Diagnostic Module (Cooke, Jernigan, Maiorana, and Myers, 1990).

During initialization, data can be entered in the functional assessment module automatically and manually. Automatic data collection loads system specific data files from the CDM and allows downloading of system health information from an aircraft data bus. At present, the operator performs manual data entries such as symptoms, parts and test equipment availability, critical states, and aircraft configuration. In the future, several of these entries may also be automated

Table 1. Hierarchical Fault/Symptom Logic

Radar System Example		
Maintenance Level	Symptom	Faults
On-Equipment	Radar Inoperative	Transmitter Bad Receiver Bad Antenna, No Sweep[a]
Off-Equipment	Antenna, No Sweep	Motor Frozen Servo Inoperative Power Circuit Bad[a]
Off-Equipment	Power Circuit Bad	Transformer Resistor Capacitor

Note.
a. This fault is isolated through testing.

through links to the Core Automated Maintenance System (CAMS), supply computers, and so on. Figure 1 shows the sequencing of algorithms and analyses performed by the initialization process and functional assessment module.

After initialization the functional assessment module uses automatic and manual data input to evaluate fault combinations and to rank tests. The module then compares tests, by time analyses and failure probabilities, to repair or replace activities; thus, it identifies the action with the highest likelihood of fixing the problem in the least amount of time. Three lists of ranked tests and/or rectifications are developed and can be selected and presented to the maintenance technician: (a) ranked tests, (b) ranked rectifications, and (c) interleaved tests/rectifications. Although a "best" action is recommended, the technician can choose any of the listed options. When the technician selects a test or rectification/maintenance action, the presentation system displays technical order instructions for performing the selected activity. If the selected action is a test, the functional assessment module evaluates the status of the faults based on the test outcome and evaluates available options. If the selected action is a rectification or maintenance action, the functional assessment module reinitializes the fault/symptom status using results obtained from a functional check. This procedure continues until the fault is isolated and the system is repaired.

II. IMIS DIAGNOSTIC MODULE (IMIS-DM) DESIGN

IMIS Controller

The IMIS controller is an executive system that controls and manipulates three subsystems: (a) an applications system (such as IMIS-DM, pre/post flight, phase inspection, and weapons load), (b) the data base module, and (c) the presentation system. Figure 2 illustrates the system. The applications system is identified as the diagnostic module.

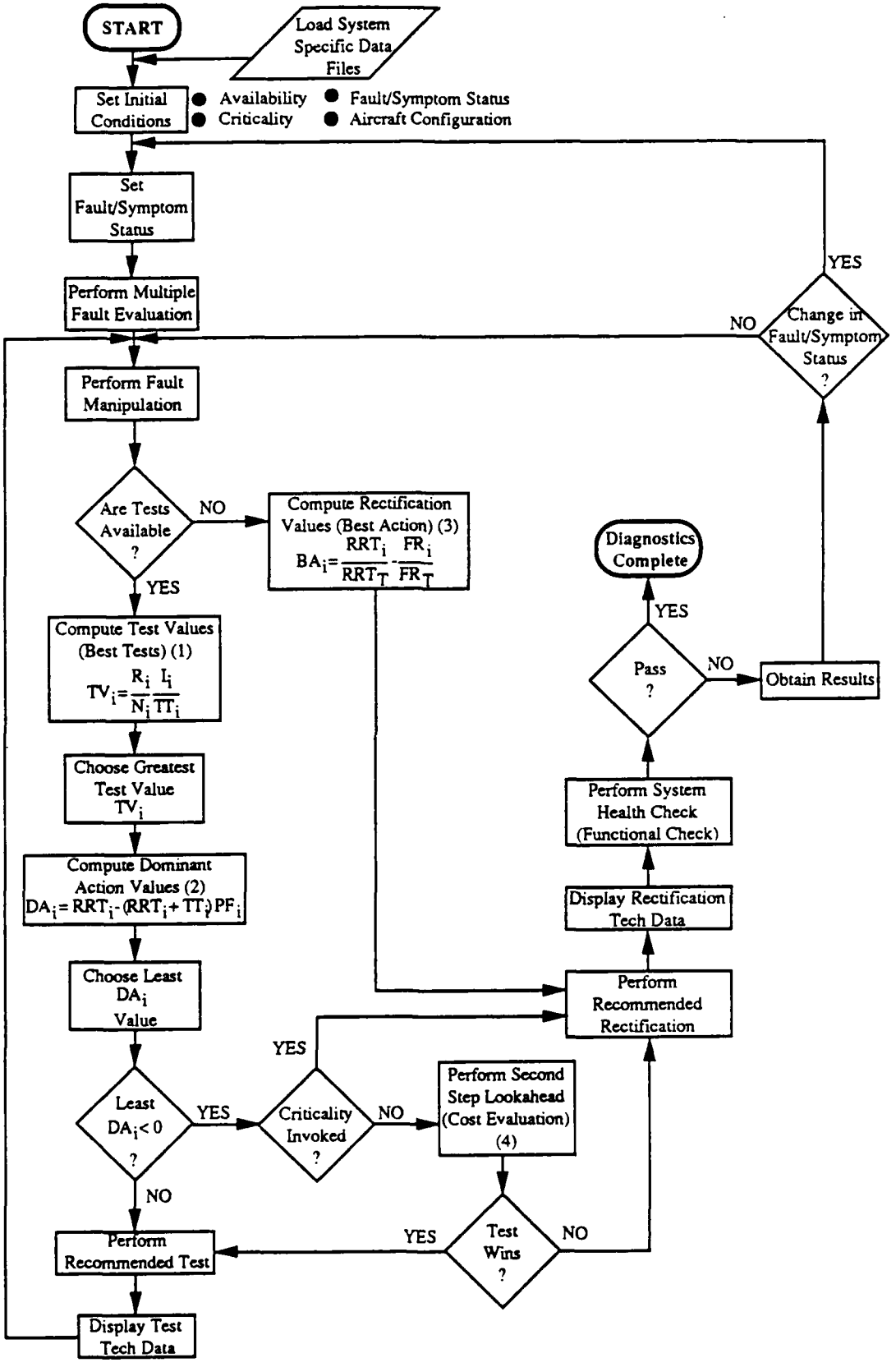


Figure 1. Logic Flow.

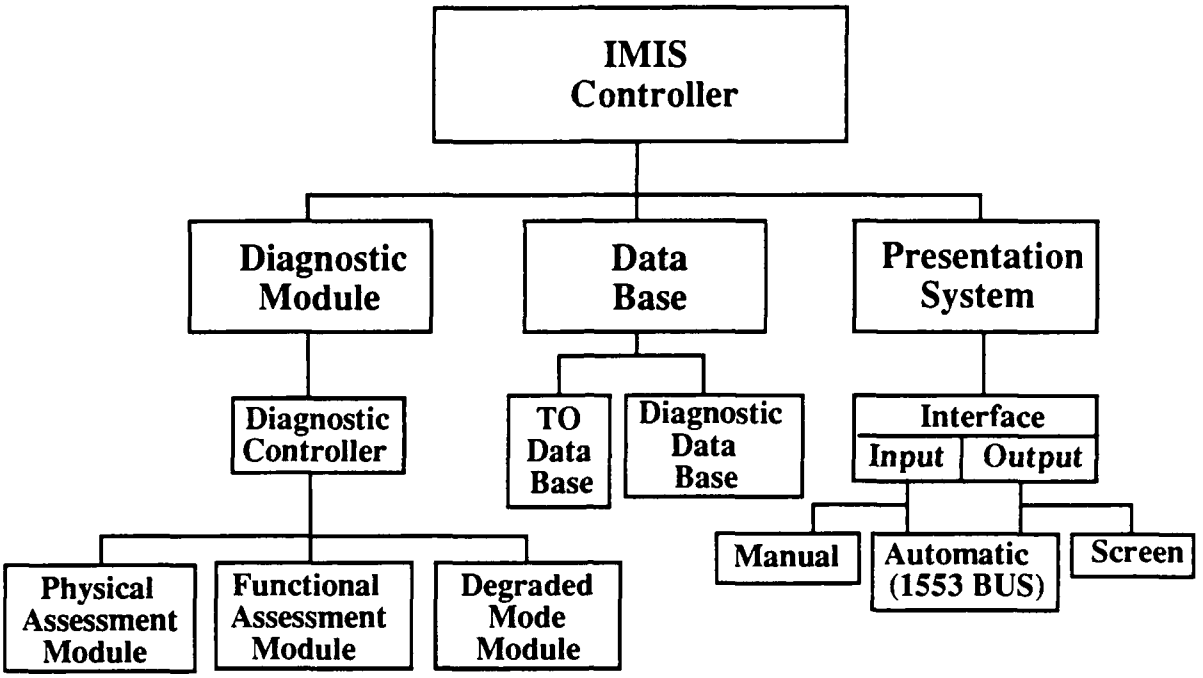


Figure 2. IMIS Controller System.

The diagnostic module is comprised of four major submodules: (a) the diagnostic controller, (b) the physical assessment module, (c) the functional assessment module, and (d) the degraded mode module. The diagnostic controller module regulates data items, diagnostic groupings, performance of its submodules, and interfacing to the IMIS controller. Interfaces between the IMIS controller and the diagnostic controller provide the means to extract diagnostic data from the data base and present information to the presentation system. The functional assessment module was described briefly in the background; the remaining diagnostic modules are new developments and are described in this section.

The data base module, in CDM hierarchical format, contains both Technical Order (TO) and diagnostic data information. The TO information consists of procedural text, graphical illustrations, and data element information; the diagnostic data contains mapping and probabilistic data for faults, tests, symptoms, and rectifications. Diagnostic data are used by the IMIS-DM; the TO information is used mainly by the presentation system.

The presentation system provides interfaces for aircraft system health checks, maintenance technician input, display of diagnostic module results, technical data for maintenance tasks, and system specific graphics. System health checks from MIL-STD-1553 data bus downloads and maintenance technician input give the diagnostic module pertinent information about the aircraft under investigation, logistic constraints, operational constraints, and task performance. The diagnostic module gives the presentation system fault isolation and repair information. The data base module supports both the diagnostic module and the presentation system.

Smalltalk/V Module Development

The first diagnostic module was written using the C programming language. We converted this module from C to Smalltalk/V. There are two reasons why Smalltalk was selected as a development language. The first is the rapid prototyping capabilities available through Smalltalk and facilitated through the Smalltalk environment. This environment contains most of the low-level

functions used in software development. The environment also allows a programmer to compile and test smaller pieces of code within the environment. Moreover, the Smalltalk environment allows the code to be reused. The second reason for selecting Smalltalk is for ease of integration into the presentation system.

Smalltalk is a high-level, OO programming environment. The basic building blocks in the Smalltalk environment are objects. Objects can be as simple as numbers, strings, and arrays, or they can be as complex as menus, data base managers, and diagnostic modules. Everything is an object in Smalltalk. Objects can store data in variables, perform functions, or both. All objects are types or instances of some Smalltalk class. An instance is an object described by a particular class type. A Smalltalk class describes the variable names and the functions for instances of itself. The variable names, also known as instance variables, are descriptive tokens where objects store data. The data stored in the instance variables are other objects. The class also describes the methods for its instances. Methods have the names of actions or functions that objects can perform. The method description describes each method's function in terms of Smalltalk code. The method names are passed to the object as messages to perform a particular action.

Diagnostic Module Controller

The diagnostic module controller uses hierarchical data from the CDM data base. The controller creates multiple diagnostic groups based on interdependent symptoms. These groupings are based on faults relating to common tasks (tests or rectifications). Each diagnostic group is independent and the status of the faults (plausible, exculpated, rectified, and so on) is stored in the diagnostic group to which it belongs.

To begin the diagnostic process, symptoms are passed to the diagnostic module controller by the IMIS controller. Each symptom maps to, or spans, a set of faults. Furthermore, these spanned faults can map to a set of subfaults, forming a fault tree. The lowest level faults in the fault tree are evaluated to find tests and rectifications pertaining to the original symptom. Diagnostic groups are then created by categorizing symptoms with common tests and rectifications. Symptoms that do not possess common tests and rectifications are considered independent and are categorized into separate diagnostic groups.

Physical Associations Model

The diagnostic module, originally designed to evaluate fault isolation and repair alternatives from almost a purely functional standpoint, has been enhanced to perform both functional and physical assessment. When diagnostics are approached from a purely functional standpoint, we cannot adequately address events causing malfunctions of other components, or malfunctions caused by a nearby physical event. For example, a technician may enter a compartment of an aircraft and observe that hydraulic fluid has leaked all over the bay causing a failure in a Line Replaceable Unit (LRU). Repair of that LRU would not be appropriate until the hydraulic line is repaired and the bay is cleaned. Many external causes of the functional problem (i.e., aircraft battle damage, bird strikes, environment extremes) could create problems with the sys.em under investigation. Hence, some physical model should be developed to produce an efficient cause and effect or physical association isolation and repair strategy.

In order to work with physical associations, we need to look at what makes a physical association reasonable. One key element is physical proximity. However, proximity is not enough by itself. There must be a physical event occurring that can affect systems, components, or parts near the event for a valid physical association. A physical event in this context implies that some

foreign agent can act externally to the affected component and cause a failure. This implication, then, implies there must be a source of the foreign agent and the component(s) in the vicinity must be vulnerable to damage by that agent.

If we only consider the battle damage source, there are many assessment models available that may prove more effective for this limited role than this modified diagnostic module. However, if we consider other sources of damage, such as heat, damaging liquid contamination, and high vibration levels, we can look inside the weapon system to find potential sources of the damaging physical agents. When we look at the weapon system for these sources, we define the limits of the universe of possible physical associations. Most normal physical hazards associated with operating in an airborne environment are already built into the Mean Time Between Failures (MTBF) and the fault weightings considered in the functional diagnostic module (e.g., routine g-loads, normal vibration levels, operating temperature extremes, and humidity). Consequently, the physical associations model must address hazards outside the range of "normal" hazards associated with operating in an airborne environment.

Therefore, the hazards to which aircraft parts and components may be subjected are few. Among these are (a) temperature extremes, specifically high temperatures; (b) liquids such as fuel, lubricants, hydraulic fluid, water, and so on; and (c) physical abuse. Physical abuse is the most widespread category because it includes both internal and external sources and has a wide range of potentially severe effects. These sources can be internal (explosion of Cartridge Activated Device (CAD); rupture of pressure vessels; slow burning/misfire of CAD; and loss of containment of high energy, spinning devices) or external (dropped objects, bird strike, mid-air collision, Foreign Object Damage (FOD), and battle damage).

Physical Effects Mapping

With a developed restricted hazards list we must identify, within some boundary (e.g., an avionics bay, an engine bay), each of the components containing hazards to either itself or to other components within the boundary. Finally, we must identify those components within the boundary that are vulnerable to these hazards. Vulnerable, in this case, refers to the functional model and implies some component may not operate within prescribed functional limits because of the effect created by a hazard normally contained within some other component.

This problematic relationship where components are vulnerable to hazards contained within some other component is a migration of the hazard. Each hazard in the restricted hazards list has individual migration traits and must be mapped according to the particular hazard. For instance, high temperature hazards tend to manifest themselves where the heat is exhausted and in the upper area of a bay; whereas, fluids tend to migrate with gravity to the lower areas of the bay. Physical abuses also tend to follow the migration concept and tend to affect their components within restricted geometric bounds. The logical approach to tracking and marking boundaries or hazard areas can be described within a three-dimensional coordinate system.

Assuming data to support the above discussion are available, then the diagnostic model must be altered to consider the effects of these physical relationships. If, during a diagnostic or other maintenance task, evidence of the presence of a physical hazard is discovered, the maintenance technician is faced with two problems. First, he must identify the source of the hazard

and, if necessary, rectify the failure that released the hazard.[1] Then, he must identify and, if necessary, repair any failed components. The data to support this scheme could be represented as in Table 2.

Physical Model Operation

The logic flow for physical assessment modeling appears as in Figures 3 and 4 as a result of the above source and effects theory. This logic flow maximizes the capabilities built into the current functional assessment module and expands upon the processing and modeling schema. The branching and control mechanisms have been built into the current diagnostic module using the logic-based PROLOG of Smalltalk/V. This paper briefly describes the physical assessment module's operation. Details of the algorithms and operations in these figures can be found in the report Integrated Maintenance Information System (IMIS) Diagnostic Module Redesign (Cooke, Jernigan, Maiorana, and Myers, 1990).

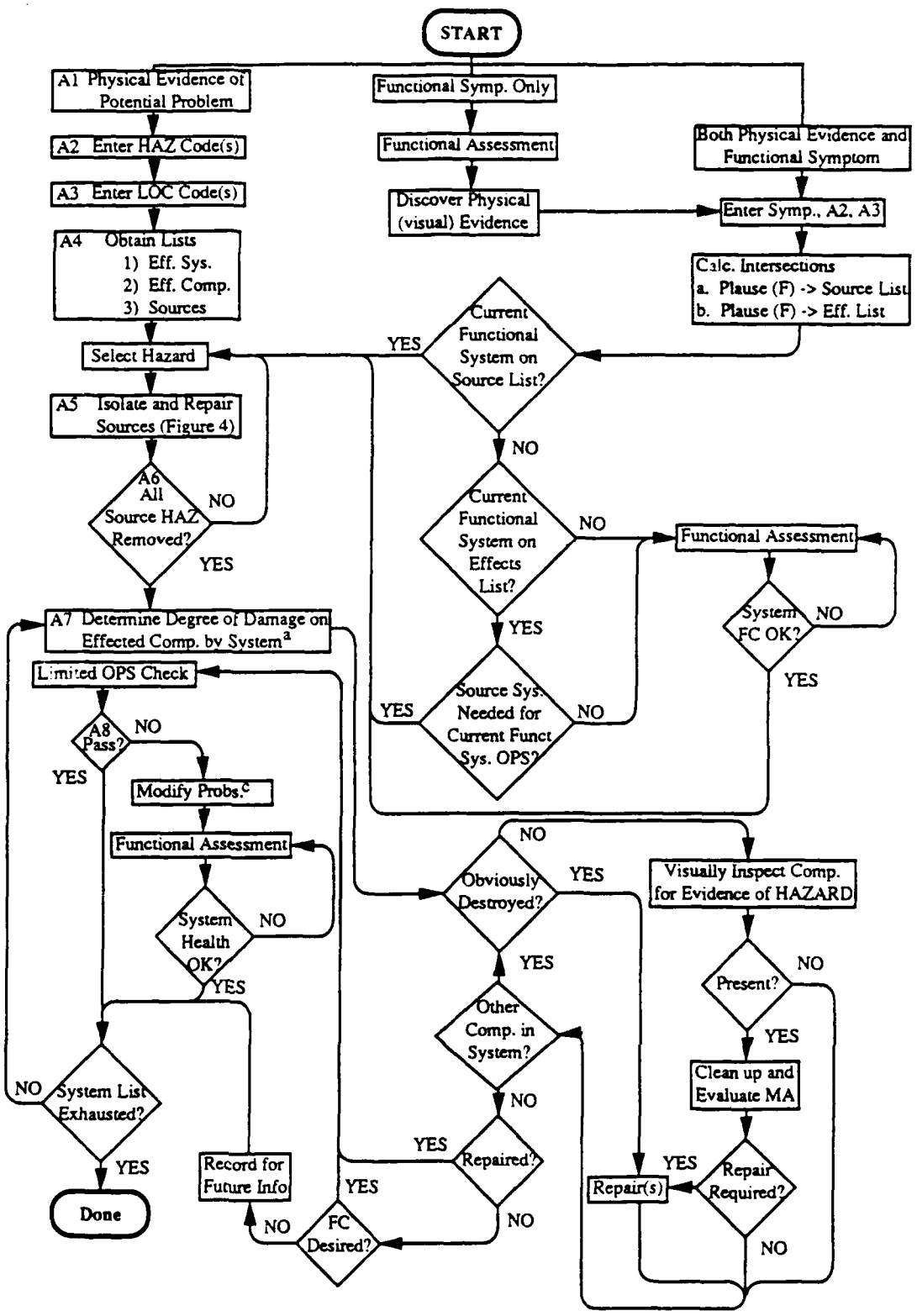
Table 2. Hazard Source and Effects Mapping

Rectification (Rect)	Hazard (H)	Vulnerability (V)	Source Fault(s) F(s)	Effect Fault(s) F(e)	Location (LOC)
A	a	-	1	-	
	b	-	2	-	
	-	c	-	1,3,4	
B	c	-	5	-	
	-	a	-	6,7,8	
C	-	c,a	-	9	
Rect ID	Hazard contained in Rect	Hazards Rect is vulnerable to	Faults which can lead to Hazard release	Functional faults which may result from exposure to Hazard	X,Y, Z Location for Rect

Upon initialization, manual and automatic system health information and physical evidence information (e.g., hazard codes, location) is entered. The diagnostic controller module then determines how to approach the diagnostic problem based on the data entries observed and diagnostic groupings available. The functional assessment approach is initiated when only functional symptoms have been observed and there is no physical evidence of a potential problem. If physical evidence appears during functional isolation and repair assessment, the diagnostic controller module redirects efforts to physical isolation and repair assessment without losing the information gained from previous actions.

The second approach, physical assessment, is initiated when physical evidence of a problem is observed, e.g., bullet hole in engine bay, hydraulic fluid in avionics bay. The physical assessment module performs diagnostics by first addressing source faults. Figure 4 provides the logic flow for alleviating source faults (A5 on Figure 3). Initially, source repair is performed on

[1] The individual technician is central to the discussions in this paper. For simplicity, we have used the singular pronoun "he" to designate the individual technician (whether that person is a man or a woman).



s = systems
c = components
Plause(F) = The set of plausible functional faults

Figure 3. Physical Assessment Model.

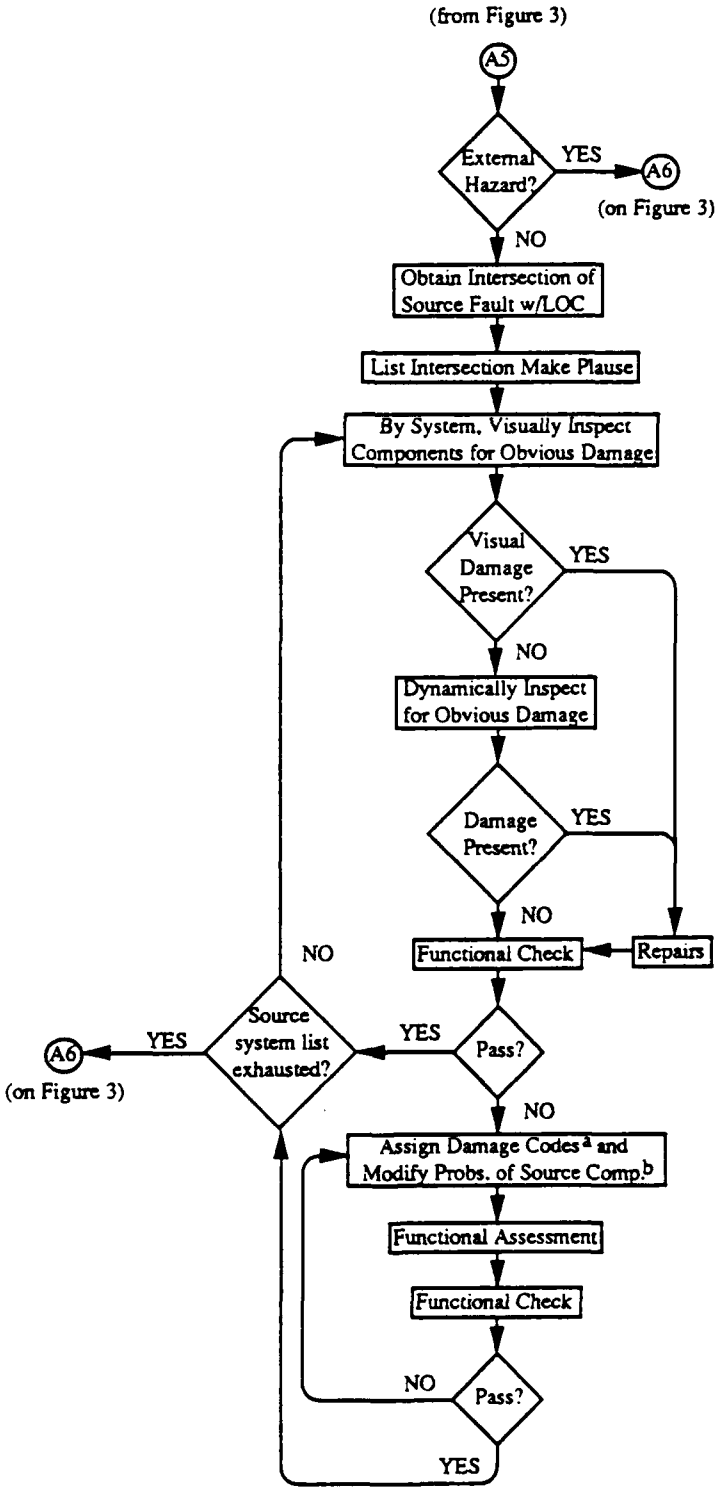


Figure 4. Isolate and Repair Sources.

Note.

[a]A description of the damage codes and formulas for modifying probabilities of components and faults are as follows:

Damage Code - (DC)

Code	Value	Desc
DS	100	Destroyed
DM	80	Damaged (dented, soaked, scorched)
SP	20	Suspected (dinged, scratched, dampened, liquid spots)
OK	1	No Apparent Effect

[b]Modifying probabilities of source components and faults. The following formulas are evaluated to obtain normalized modified source component and fault probabilities for a given location. Although formulas 3 and 4 are only listed once, both component and fault probabilities are calculated using the same formulas, but are performed separately.

$$Prob_{i,1} = \frac{\sum^{Comp} \frac{1}{MTBF\,(Fs)}}{\sum^{LOC} \frac{1}{MTBF\,(Fs)}} \quad (1)$$

$$Prob_{i,1} = \frac{\frac{1}{MTBF(Fs)}}{\sum^{LOC} \frac{1}{MTBF(Fs)}} \quad (2)$$

$$Prob_{i,2} = Prob_{i,1} * DC \quad (3)$$

$$Prob_{i,F} = \frac{Prob_{i,2}}{\sum Prob_{i,2}} \quad (4)$$

Where, DC = component damage code value,
Fs = source fault,
Comp. = component,
LOC = location,
$Prob_i,1$ = probability of the ith source component or fault,
$Prob_i,2$ = modified probability of the ith source component or fault, and
$Prob_i,F$ = normalized modified probability of the ith source component or fault.

[c]Modifying probabilities of effected components and faults. The following formulas are evaluated to obtain normalized modified effect component and fault probabilities for a given location. Although formulas 7 and 8 are only listed once, both component and fault probabilities are calculated using the same formulas, but are performed separately.

$$Prob_{i,1} = \frac{\sum_{1}^{Comp} \frac{1}{MTBF\,(Fe)}}{\sum_{1}^{LOC} \frac{1}{MTBF\,(Fe)}} \quad (5)$$

$$Prob_{i,1} = \frac{\frac{1}{MTBF(Fe)}}{\sum_{1}^{LOC} \frac{1}{MTBF(Fe)}} \quad (6)$$

$$Prob_{i,2} = Prob_{i,1} * DC \quad (7)$$

$$Prob_{i,F} = \frac{Prob_{i,2}}{\sum Prob_{i,2}} \quad (8)$$

Where, Fe = affected fault,
$Prob_i,1$ = probability of the ith affected component or fault,
$Prob_i,2$ = modified probability of the ith affected component or fault, and
$Prob_i,F$ = normalized modified probability of the ith affected component or fault.

components showing obvious physical damage, then the functional assessment module is activated to alleviate source faults that are not so obvious. The second step in the physical assessment module is to isolate and repair affected components. Again the approach to physical assessment of affected components is to first repair obviously damaged components. When all known damaged components are repaired, the diagnostic controller module reverts to functional assessment to complete the aircraft repairs.

As shown in the figures, the physical assessment module's logic flow maximizes the maintenance technician's abilities to evaluate and act upon physical evidence without being delayed by the details of the general technical data required to identify, clean up, and evaluate hazard exposure. However, the processing is available to provide additional information to assist the novice through the details if necessary.

OO PROLOG Physical Associations Module

Although Smalltalk/V offers well-developed rapid prototyping techniques within an OO environment, it also offers the opportunity for logic-based programming or PROLOG. With the development of the physical assessment module theories and operations described in Section II, a logic-based system could more closely duplicate this logical decision process designed from a maintenance technician's viewpoint.

The first step in implementing the logic-based physical assessment module was to identify and develop a mock hierarchical CDM dictionary containing all of the necessary data elements required of the physical module. This was just an extension of the newly developed functional model's mock CDM dictionary and was implemented within the Smalltalk OO environment. Initial format and mapping of the physical module's CDM elements are described in Table 3.

Table 3. CDM Requirements for Physical Associations

Rect Hazard ID	Hazard Names
Rectification	Hazard Codes
Affected Faults	Hazard Meaning
Source Faults	
Hazards Vulnerable to	Damage Names
Hazard Names	Damage Codes
Hazards Contained	Damage Meaning
Hazard Names	
System Name	System Name
	System Code
	System Meaning

After implementing the above CDM elements, we identified the constraints of the Smalltalk/V PROLOG environment. The major constraint imposed was that Smalltalk/V PROLOG had no input or output functions and all user and data interfaces had to be developed within the Smalltalk/V OO environment to test the accuracy of the physical module. Therefore, Smalltalk/V OO Physical Presentation Module (PhysPresModule) and Physical Model categories were developed to help interface with the functioning and testing of the logic-based physical assessment module.

The PhysPresModule simulates the CDM data capture and manipulation by the IMIS controller and the user interfacing of the presentation system. Two classes, the PhysController and the PhysModel category, were created to implement the Smalltalk PROLOG version of the physical assessment module. The PhysModel class describes the rules and functions needed to perform the logical diagnostic decision process of the physical assessment module. This class is a subclass of the PROLOG class, which is a subclass of Logic in the PROLOG-V category. Once lists of pertinent data are passed from the PhysController class to the PROLOG data base, rule firing dictates the logical approach of physical diagnostics as described in Section II. If any information or user interface is required, the PhysModel simply uses Smalltalk/V commands to access and perform methods within the PhysPresModule. Responses are then passed to the PROLOG dictionary. The PhysController class describes the instance variables and functions needed to perform and interface the PhysModel of Smalltalk/V PROLOG. The main objective of this class is to act as an interface between the Smalltalk/V MDAS controller and the PROLOG physical assessment module.

Degraded Mode

Degraded mode occurs when the diagnostic module can no longer recommend an action and all suspected faults, given symptom occurrence, have been eliminated from consideration. This situation can happen if the diagnostic module is given incorrect or incomplete data. At this point, normal diagnostics are suspended and the diagnostic module is in the degraded mode once the technician approves this action.

The objective of the degraded mode is to have a test fail, resulting in a plausible set of faults, and/or do a rectification that passes the system health. In certain situations, the technician may choose to put the system into degraded mode in one of two ways. One, the technician can select degraded mode if he decides the diagnostic module is no longer helping in troubleshooting. Or, in response to the recommendation of the diagnostic module, he can revert to the degraded mode.

The diagnostic module will recommend degraded mode of operation if:

1. a symptom is present but all suspected faults that could have caused the symptom are eliminated from consideration through passed tests or alleviation of another symptom; or,

2. if rectifications have been performed for the second time on the same components where the fault is suspected; or,

3. a symptom is confirmed present and is not in the data base.

Although the maintenance technician has full control of degraded mode selection, degraded mode is necessary to continue diagnostics when:

1. the diagnostic module recommends the degraded mode based on the above occurrences;

2. Can Not Duplicate (CND)/Intermittent (unverifiable starting points) messages are received; and,

3. whenever the technician chooses, i.e., the technician does not want to follow any of the diagnostic module's recommendations.

When the degraded mode is selected, the physical and functional assessment modules are suspended by the diagnostic controller module and a message appears notifying the technician he has entered degraded mode. This message remains on the screen at all times during degraded mode assessment.

While in degraded mode, the physical and functional assessment modules are not able to provide the technician with any recommended actions. To aid the technician, a smart Table Of Contents (TOC) is created. The smart TOC consists of two lists. The first list contains ranked rectifications based on component MTBFs. The components with the lowest MTBFs are ranked highest on the list. The second list contains an ordered list of tests based on probability of failure, calculated by summing the failure rates of all spanned faults. Moreover, to provide a more accurate test ranking, any information gained during physical and functional assessment is used to alter the test's failure probabilities. When modifying the test's failure probabilities, exculpated faults are not used in computations of tests containing them.

Upon display of the TOC, the technician selects a test or rectification from the ranked lists and performs the selected action. Depending on the action performed and its results, diagnostics can proceed in several ways. At the completion of any action, the maintenance technician must either suspend diagnostics, select another action from the TOC, or exit the degraded mode. The technician must consider the precedences described below, reflecting logical continuation of isolation and repair.

1. If a test is selected during degraded mode assessment and a fail result is exhibited, a new plausible set is established. The degraded module then records the new plausible set of faults for further isolation and repair. Once a new plausible set of faults is established, degraded mode can be exited and physical or functional assessment can proceed from the new plausible set.

2. If a test is selected during degraded mode assessment, and a pass result is exhibited, the degraded module records the exculpated faults, eliminates the performed test from consideration, and reranks the TOC's test list. The maintenance technician can then select and perform another action from the TOC.

3. If a rectification is selected and performed during degraded mode assessment, and changes to the system are exhibited as a result of a functional check, the degraded module records the appearance of new symptoms and/or existing symptoms are modified or deselected. Given the new set of observed symptoms, degraded mode can be exited and diagnostics can proceed with physical or functional assessment.

4. If a rectification is selected and performed during degraded mode assessment, and no changes to the system are exhibited as a result of the functional check, the degraded module records that rectification, eliminates the performed rectification from consideration, and reranks the TOC's rectification list. The maintenance technician can then select another action from the TOC.

Hence, the diagnostic module can continue fault isolation and repair when faced with incorrect or missing data.

III. DIAGNOSTIC ENHANCEMENTS

Enhanced Diagnostic Module Functions

The enhancements described in this section have created a more efficient and accurate diagnostic module. The module now performs degraded mode and critical fault assessment, and captures information gained from previously failed tests. It also considers dependent symptom occurrence and time saved for accessing groups of components or LRUs for testing and repair. Other enhancements to the diagnostic module allow for changes to test and functional check outcomes in case of incorrect entry of results.

Failed Faults from Previous Test

Under unusual circumstances, earlier versions of the diagnostic module could lose fault isolation information due to fault combination manipulations. A new type of fault list is used in the enhanced version to correct this problem. This new list type is called the isolated faults list. Whenever the plausible fault list contains only one fault, the fault is placed in the isolated faults list before other processing is done. When the plausible set is rebuilt, the isolated faults list is searched for the first isolated plausible fault. If one is found, all other faults are moved to the maybe set. Because the fault is the only one in the plausible set, the module will recommend its rectification with a 100 percent probability.

Access Group

An access group is a group of components unveiled by removing a panel or cover. When ranking tests or rectifications, a diagnostic advisor should consider access group factors for rectification and testing time efficiency. Diagnostic efficiency may be gained when actions performed on access groups reveal more fault-associated components but have high access times. Previous IMIS-DM versions did not consider access times in the ranking of tests and rectifications. Access times in previous versions were assigned to each individual action and were not considered for a commonly accessible group of actions.

The method of approach used to develop this capability was essentially the same as that used to develop a Multiple Outcome Test (MOT) evaluation capability described in the paper Integrated Maintenance Information System (IMIS) Diagnostic Module (Cooke, Jernigan, Maiorana, and Myers, 1990). The access group algorithm is designed so that once access is gained, each test in the group can be accomplished as though no access time is required. In

addition, the best test evaluation in these circumstances is merely an extension of the current best test algorithm. This feature was created by adding an enhancement factor to the best test that accounts for the additional fault isolation capability obtained by gaining access.

The enhancement factor used is:

$$EF_j = \frac{\sum_{k=1}^{N}\left[\frac{\sum_{i=1}^{PS} FR(1)}{FR(PS)} \times \frac{\sum_{i=1}^{PS} FR(0)}{FR(PS)}\right]}{N \times \Sigma T}$$

Where, EF_j = the enhancement factor for test j,

$FR(PS)$ = the sum of all the failure rates for faults in the plausible set,

$\sum_{i=1}^{PS} FR(1)$ = the sum of the failure rates for spanned faults for a given test (T) in the plausible set of faults,

$\sum_{i=1}^{PS} FR(0)$ = the sum of the failure rates for unspanned faults for a given test (T) in the plausible set of faults,

$\sum_{k=1}^{N}$ = the sum of the products of spanned and unspanned tests within the access group,

ΣT = the sum of all test times including access time for the group, e.g., for an access time of 10 minutes creating access to three five-minute tests, $\Sigma T = 25$, and

N = the number of tests in the access group.

The enhancement factor is set to zero if no additional tests are included in the access group. Hence, the best test algorithm used in the IMIS-DM after this enhancement is shown below and specific formulas for individual variables can be found in the report referenced above.

$$BT = \max \frac{R_j \bar{I}_j}{T_j} + EF_j$$

Where, BT = Best Test value,
R_j = sparseness ratio of the test span,

$\bar{I}_j$ = the average information gain, and
T_j = time to accomplish test j.

"But Not" Data Entry

The "But Not" algorithm implemented in the redesigned diagnostic module retrieves information from test results in the form of observed outcomes and spanned faults, and determines what faults are implicated and exculpated based on the test performed and outcome observed. Previous versions of the diagnostic module did not include "But Not" data entry logic when manipulating faults from test results. The exclusion of this "But Not" data entry logic resulted in inefficient fault isolation and repair decisions because faults known to be good remained under investigation.

The "But Not" algorithm development approach first identified what test outcomes would be returned from test results and how faults would be manipulated. Every test has one pass outcome and at least one fail outcome. Pass outcomes only exculpate faults and if a pass outcome is observed, no fail outcomes can be exhibited. Fail outcomes, however, can implicate and exculpate faults simultaneously.

There are two types of tests the "BUT NOT" algorithm operates on: BINary (BIN) tests and MOTs. The BIN tests are very simple tests that exhibit either a pass or fail outcome, while the MOTs are more complex. MOTs have one pass outcome and two or more fail outcomes. There are three types of MOT tests: (a) Complete And Enter One (CAEO), (b) Complete And Enter All (CAEA), and (c) Exit At First Failure (EAFF). The CAEO MOT is completed in full and only one outcome can be entered upon completion of the test. However, CAEA MOTs are also completed in full but all observed outcomes are entered. EAFF MOTs are only completed to the point at which the first failure is observed and at that point the observed outcome is entered.

Each outcome exhibited from a test result maps to a set of spanned faults, exculpated and/or implicated. BIN, CAEO, and CAEA tests have one pass outcome that, when observed, exculpates all faults spanned by the pass outcome. When a fail outcome(s) is observed from these tests, the diagnostic module implicates and exculpates all faults for the observed fail outcome(s) and then exculpates all the implicated faults of the non-observed outcomes. The EAFF also exculpates all faults of the observed pass outcome. But, if a fail outcome is observed the "But Not" algorithm exculpates all implicated faults for prior non-observed outcomes in the performed sequence and implicates and exculpates faults of the observed outcome. Because of the implementation of the "But Not" data entry logic, known good faults are exculpated while suspected faults are isolated and repaired.

Account for TOC Actions

Earlier versions of the diagnostic module did not effectively allow choices to be made from the TOC. This limited the maintenance technician's ability to perform tasks he considered pertinent but were not in the interleaved actions list. Accounting for TOC actions was implemented easily within the diagnostic module. Now, whenever a TOC test or rectification is chosen, the diagnostic module only needs to be informed the action selected was not from the interleaved actions list. If a test is selected, the observed test outcomes need to be passed to the diagnostic module as well. Furthermore, if the task pertains to any of the existing diagnostic groups, that group will be updated. If no appropriate diagnostic group exists, one is created.

Change Test Result

If a maintenance technician erred in the selection of a test outcome, earlier diagnostic module versions would not allow correct entries to be made easily. To ease changing a test's result, the previous actions list is used. The previous actions list contains a list of all the tests and rectifications previously performed. It also keeps a copy of the machine's diagnostic state before the action. Once a previous test is selected, the new results are passed to the diagnostic module. The diagnostic group pertaining to that test is removed and replaced with the copy stored with the previous action. If no diagnostic group exists, the copy stored with the test is added to the list of diagnostic groups. The new test results are passed to the diagnostic group. The next item in the previous actions list belonging to the same diagnostic group as the test is updated with the new diagnostic group. Then the diagnostic group is updated with the results of the action. This process is repeated until the previous actions list is exhausted.

Change Functional Check Result

Previous versions of the diagnostic module would not allow a maintenance technician to change manually entered functional check results. If a maintenance technician erred when selecting functional check results, the diagnostic module would continue diagnostic evaluation with incorrect system information and proceed to an incorrect isolation and repair decision. The previous actions list is accessed so the technician can change functional check results. The previous actions list contains a list of all the functional checks, tests, and rectifications previously performed by the maintenance technician, and records a copy of the machine's diagnostic state before each action. Once a previous functional check is selected, the new results are passed to the diagnostic module. The diagnostic group pertaining to that functional check is removed and replaced with the copy stored with the previous action. If no diagnostic group exists, the copy stored with the functional check is added to the list of diagnostic groups. The new functional check results are passed to the diagnostic group. The next item in the previous actions list belonging to the same diagnostic group as the functional check is updated with the new diagnostic group. Then the diagnostic group is updated with the results of the action. This process is repeated until the previous actions list is exhausted.

Revised Criticality

A criticality function has been designed for situations when operational demands prevent maintenance practices that take a minimum time to repair a malfunctioning system. Under some circumstances, operational requirements would declare a system usable if it can be determined the fault present in the system is in a part of the system not essential for the next sortie. The diagnostic data allow faults which may or may not be designated as critical to be assigned to certain systems and subsystems. If a system or subsystem has been declared critical, then the diagnostic module is modified to make recommendations based upon a criticality algorithm.

In developing the revised criticality algorithm, we modified the definitions of critical tests and rectifications to include multiple fault scenarios. The revised definitions are:

1. Critical Test - A test that examines all potential faults declared critical.

2. Critical Rectification - A rectification that acts to repair all faults declared critical.

Based on these definitions, we developed a revised criticality algorithm to allow a critical fault decision at the earliest possible time in the maintenance process. The revised criticality algorithm now utilizes information theory to calculate a value for steps to isolate and repair

(Equation 1), steps to operationally ready or fly (Equation 2), and total steps to repair (Equation 3). Information theory shows the minimum number of steps to fault isolation in a given set of faults is expressed by the base 2 logarithm of the number of faults in the plausible set (Log_2 #F).

The values from Equations 1-3 are then used to evaluate advantages to operational considerations (Equation 4) and cost to total maintenance steps (Equation 5), an isolation and repair decision is then made based on the revised criticality formula (Equation 6). The revised criticality algorithm formulas expressed below are a summary of investigations and theories employed in the development of the revised criticality function. Details of the algorithms can be found in the Integrated Maintenance Information System (IMIS) Diagnostic Module Redesign (Cooke, et. al., 1990).

$$STREP = \{\Sigma[Pfns * Log_2(2*\#fns)]\} + \{\Sigma[Pfs * Log_2(2*\#fs)]\} + 1 \quad (1)$$

$$STFLY = \{\Sigma[Pcf * Log_2(2*\#cfs)]\} + 1 \quad (2)$$

$$STTOT = \{\Sigma[Pfns * Log_2(2*\#fns)]\} + \{\Sigma[Pcf * Log_2(2*\#cfs)]\} + 1 \quad (3)$$

Where, STREP = the steps required to repair a system when the most appropriate test is performed first,
STFLY = the steps to declare a system ready for operations when a critical test is performed as the first test,
STTOT = the steps to perform maintenance required to bring the system to operational condition after a critical test,

ΣPfns = the sum of the probabilities of the non-spanned faults,
#fns = the number of non-spanned faults,
ΣPfs = the sum of the probabilities of the spanned faults,
#fs = the number of spanned faults,
ΣPcf = the sum of the probabilities of the faults in each test outcome that implicates critical faults, and
#cfs = the number of faults spanned in each test outcome that spans critical faults.

The advantage to operational considerations is:

$$ADVOPS = STFLY - STREP \quad (4)$$

The cost to total maintenance steps is:

$$STMAIN = STTOT - STREP \quad (5)$$

The decision equation for whether the critical test should be preferred over a conventionally chosen test is then expressed as:

When $Val*ADVOPS \geq STMAIN$, do the critical test. (6)

Where, Val = the value of an operational hour over a maintenance hour,
ADVOPS = the advantage to operations of performing a critical test first, and
STMAIN = the cost of total maintenance steps.

Enhanced Diagnostic Presentation Capabilities

The enhancements described in this section have created a more efficient and accurate diagnostic presentation environment for the maintenance technician with a data validity check on test outcome information to control entry of incorrect or out-of-bounds test values and a feedback entry for the maintenance technician to indicate an unsuccessful maintenance action.

Data Validity Check

The maintenance technician must enter the results of a test. In previous versions, no test data validity checks were available to the technician to designate whether a particular test result was within the expected values of an acceptable test result. For instance, a particular voltage check (test) on a wire within a wire bundle should result in 5 +/- .01 volts for a pass and 0 to 4.989 volts for a fail, indicating the acceptable range of test voltage values is between 0 and 5.01 volts. In previous versions, if the maintenance technician tested the incorrect wire in the bundle and returned a value of 6 volts from the check, the options would be to pass or fail the test. After making a test entry, the presentation module would accept the word of the technician that the test was performed properly and the results were correct. This test entry was then provided to the diagnostic module with no test data error checks and in doing so, diagnostics proceeded down the wrong path to fault isolation and repair.

Hence, as a result of this investigation, the diagnostic presentation system was equipped with a data validity check, which retrieves pertinent test data values from the data base and requires a value from the maintenance technician that is within the expected realm of the test outcome.

Feedback Entry

Maintenance actions are rectifications that do not remove and replace (R&R) components but rather adjust components. Previous versions of the presentation module treated each maintenance action as an R&R and did not consider instances when a repair or R&R would be required if the maintenance action was unsuccessful or could not be performed properly. For instance, consider what would happen if a maintenance action on a component could not be performed successfully and the presentation module only acknowledges R&Rs. First, the presentation module would require a functional test even though nothing was fixed. Then, the diagnostic module, being given incorrect information on the outcome of the maintenance action, could suggest another maintenance action on the same component, perform a R&R on another component ignoring the faulty one, or perform further unnecessary tests on the faulty component or other components.

A maintenance action, when used as a rectification, presents the unique situation of a passed test requiring a system health check. The reason is the maintenance action, if successful, was a rectification (with a system health mapped as its conformation test). But, if the maintenance action was unsuccessful it mimics a test that implicates a set of faults. An example, as illustrated in Table 4, is a system with potential faults of Out Of Alignment and Will Not Align among its set of manifested failures. The Out Of Alignment requires an alignment maintenance action while the Will Not Align requires an R&R. If the align rectification is accomplished, its success must be determined before proceeding. The test mapped to the Out Of Alignment is, "was the maintenance action successfully completed?" If the answer is yes, then the system health check must be performed to ensure the Out Of Alignment fault was the fault present in the system and the alignment did in fact remove the Out Of Alignment fault. If the answer is no, then the fault Will Not Align is implicated and diagnostics and repairs associated with its set must be performed. A third option occurs if the alignment was started but could not be completed. In this case, both the

Out Of Alignment and Will Not Align faults are implicated and the repair actions required by these faults are indicated. Hence, each of these outcomes obviously produces different sets of implicated and exculpated faults and system health information.

Table 4. Maintenance Action Test Example

Select the output that represents the results of the maintenance action.			
	Completed?	Successful?	
Outcome #1	yes	yes	(pass)
Outcome #2	yes	no	(Fail 1)
Outcome #3	no	---	(Fail 2)
Outcome #1	Exculpated both faults -- Out of Alignment and Will Not Align		
Outcome #2	Implicates fault -- Will Not Align		
Outcome #3	Implicates both faults -- Out of Alignment and Will Not Align		
The maintenance action is mapped to Out of Alignment while the repair is mapped to both Out of Alignment and Will Not Align.			

Therefore, the logic to handle this unique situation requires data element modifications and presentation software modifications. Within the CDM rectification data elements for maintenance actions, the author is required to list both tests (aligned and system health) in sequence to prove that the Out Of Alignment fault was at fault and that the maintenance action did fix the problem. The first test is an MOT similar to the example below or a binary test. The second is the system health and is recommended if the first test passes.

IV. CONCLUSIONS

The IMIS-DM and presentation module provide the routines and interfacings that integrate many diagnostic theories to effectively and efficiently fix aircraft systems. These theories provide the means for incorporating information into the overall diagnostic decision making process and utilizing the information to its fullest extent when analyzing, displaying, and fixing the current problem.

Because of R&D efforts in the diagnostic arena, the diagnostic module can assess improper system behaviors caused by physical events resulting from failures of source components and causing malfunctions in other components or affected components. These events would be caused by temperature extremes, a release of fluids, and physical abuses including aircraft battle damage. One of the most important efforts to upgrade the diagnostic module involved an object-oriented redesign of the C-programmed diagnostic module (now called the functional assessment module). This redesign effort created a diagnostic module compatible with the hierarchical structure employed by the CDM and created a system that provides rapid prototyping for future diagnostic enhancements and the interfacing necessary for the physical assessment module previously described.

Enhancements as a result of previous research have allowed the diagnostic module to function effectively under less than adequate data availability conditions with the implementation of the degraded mode module. Other changes have provided tools to the technician to change the results of erred test and functional check results and to select actions from the TOC. Revised criticality, failed faults from previous tests, access groups, and "But Not" data entry have added time to repair efficiency by including more maintenance environment information and better utilization of all information during the diagnostics decision process.

Enhancements to the presentation module include the data validity check to inspect test values returned by the maintenance technician in case incorrect procedures were followed during testing or incorrect measurement was taken and the implementation of an effective means to approach maintenance actions. Previous versions of the presentation module processed maintenance actions as rectifications. The current version of the diagnostic module provides for successful maintenance actions requiring functional check performance and unsuccessful maintenance actions requiring rectification actions.

REFERENCES

Air Force Human Resources Laboratory. (1989). Draft Specification for Digital Technical Information. Air Force Human Resources Laboratory.

Cooke, G.R., Jernigan, J.H., Maiorana, N., & Myers, T.A. (1990). Integrated Maintenance Information System (IMIS) Diagnostic Module (AFHRL-TP-90-13). Wright-Patterson AFB, OH: Air Force Human Resources Laboratory.

Cooke, G.R., Jernigan, J.H., Maiorana, N., & Myers, T.A. (1990). Integrated Maintenance Information System (IMIS) Diagnostic Module Redesign (Contract No. F33615-88-C-0004). Wright-Patterson AFB, OH: Air Force Human Resources Laboratory.

LIST OF ABBREVIATIONS

AFHRL	-	Air Force Human Resources Laboratory
BIN	-	BINary
CAD	-	Cartridge Activated Device
CAEA	-	Complete And Enter All
CAEO	-	Complete And Enter One
CDM	-	Content Data Model
CND	-	Can Not Duplicate
DC	-	Damage Codes
EAFF	-	Exit At First Failure
FOD	-	Foreign Object Damage
IMIS	-	Integrated Maintenance Information System
LRC	-	Combat Logistics Branch, AFHRL
LRU	-	Line Replaceable Unit
MDAS	-	Maintenance Diagnostic Aiding System
MIL-STD	-	Military Standard
MOT	-	Multiple Outcome Test
MTBF	-	Mean Time Between Failures
OO	-	Object Oriented
PROLOG	-	Programming in Logic
R&D	-	Research and Development
R&R	-	Remove and Replace
TO	-	Technical Order
TOC	-	Table Of Contents

GLOSSARY

Action. A diagnostic or corrective procedure performed by a maintenance technician.

Aircraft Configuration. Placements or layouts of aircraft system components.

Availability. A component's or test equipment's obtainability for use in the diagnostics process.

Best Rectification. A multiple fault algorithm that chooses the optimum action from among available rectification actions.

Best Test. A diagnostic software algorithm that chooses the optimum test from among those available at any point in the diagnostic sequence.

Component. The lowest physical level of indenture on which a maintenance technician at a given level of maintenance (i.e., organizational, intermediate, and depot (O, I, D)) will normally work. For example, an organizational level maintenance technician would consider a Line Replaceable Unit (LRU) as a component; whereas, an intermediate level technician would consider the LRU an end item and the Shop Replaceable Unit (SRU) a component.

Critical Rectification. A rectification that acts to repair all faults declared critical.

Critical Test. A test that examines all potential faults declared critical.

Criticality. A measure of need for a particular system capability. For example, a fault in an air-to-ground function might not be critical for an air defense sortie, whereas a fault in an air-to-air function would be critical for the same sortie requirement.

Dominant Action. A rectification action whose likelihood of success is so great it is recommended before available tests that would reduce the plausible set.

Failure Rate. The inverse of Mean Time Between Failures (MTBF).

Fault. The cause of an equipment malfunction. The manifestation, through either inference or direct observation, of a failure within a system.

Feedback Analysis. The process of collecting parameters while in the maintenance/diagnostic environment and using these parameters to update current logistics information.

Feedback Loop. An interconnection of faults and signals such that no single test point can successfully isolate the fault location.

Functional Check. A test performed to ensure a rectification action has been successful in restoring a system to operational status.

Maintenance Action. A rectification that does not involve removal and replacement of a component, but is merely an adjustment.

Mean Time Between Failures (MTBF). The unit of reliability used as a predictor of fault likelihood. Its inverse is the failure rate.

Multiple Faults. An event where two or more faults (failed components) exist simultaneously in a given system.

Multiple Outcome Test (MOT). A test procedure without a binary pass/fail result. The procedure may have any number of outcomes; however, each outcome is unique and distinguishable from all other outcomes.

Plausible Set. The set of possible faults that could logically have led to an observed or indicated faulty condition. The elements in this set of faults contain single faults or combinations of faults that are not redundant.

Rectification. The repair of a fault(s) which alleviates a symptom or set of symptoms.

Repair Time. The time required to complete system repair after a fault is isolated. It may include access times. It will include reinstallation of original components removed unnecessarily as part of diagnostics, secure and closure, and final functional check.

Support Equipment. Tools or devices needed to perform an action.

Symptom. A machine-generated or verbal description code indicating that a malfunction exists (e.g., "Receiver, no audio").

Test. A prescribed sequence of actions whose result will implicate or exonerate a set of faults.

APPENDIX: SMALLTALK DEFINITIONS

Diagnostic Module's Smalltalk Class Structure

To implement the Smalltalk version of the diagnostic module several classes were created: controller, previous action, diagnostics, sets, a combination generator, combination, tasks and ranked task. The diagnostics and controller classes are the primary classes used in the module redesign and are discussed below.

Diagnostics Class. The diagnostics class describes the instance variables and functions needed to perform diagnostics. Each instance of this class is equivalent in functionality to the diagnostic module written in C. The difference is the C version allowed only one diagnostic group, now there can be many.

Controller Class. One instance of the controller class is needed to perform diagnostics. This instance creates, monitors, and controls one or more diagnostic groups. This process is described in Section II. Furthermore, the interface into the diagnostic module is defined by the controller class. This feature gives the presentation system a focal point of communication to the diagnostic module and the ability to port the diagnostic module to different presentation systems. In the C version, this interface was spread throughout the presentation system and porting the diagnostic module was more difficult. More details on the interface are described in the Appendix.

Passing Information to The Diagnostic Module Controller

Critical Faults. Informing the diagnostic module controller of critical faults is done through the "setCriticalFaults:" message. This message requires a parameter consisting of a list of critical faults.

Observed Symptoms. To manually set observed symptoms the "addSymptoms:" message is used along with a list of symptoms.

Results of a Test. The "ranTest:withOutcomes:" message is used to pass the results of a test. This message requires two parameters. The first parameter is the instance of test that was performed. The second is a list of outcomes observed as a result of performing the test.

Rectification Performed. To inform the diagnostic controller a rectification has been performed the "performRectification:" message is used. The parameter passed along with this message is the rectification.

Changed Results from a Previous Test. The message "change Results From Previous Test: with Outcomes:" is used to change results from a previous test. The first parameter used for this method is a test element from the previous actions list. The elements from the previous actions list contain necessary information. The necessary information includes the machine's state before the test was performed as well as the test. The second parameter is the new outcomes or test results. The current machine state is replaced with the one stored in the previous action element. Now the machine state is modified with the new test results. Using other tasks performed after the test was run initially, the current machine state is updated.

Retrieving Information from The Diagnostic Module Controller

Obtaining the List of Previous Actions. To obtain a previous actions list from the diagnostic controller the "previousActions" method is used. This method returns an ordered list of tests and rectifications. The test items will also contain the observed outcomes of the test.

Obtain the Best Task. To obtain the top interleaved task from the diagnostic controller, use the "bestTask" message.

Obtaining Ranked Lists. There are three types of ranked lists obtainable from the diagnostic controller: (a) interleaved tasks, (b) ranked tests, and (c) ranked rects. The interleaved tasks list is a ranked listing of tests and rectifications based on the dominant action algorithm. To retrieve an interleaved tasks list from the diagnostic controller, use the "interleavedTasks" message. The ranked tests list is an ordered list of tests based on the best test algorithm. To obtain the ranked tests list from the diagnostic controller, use the "rankedTests" message. The ranked rectifications list is an ordered list of rectifications based on the best rects algorithm. To obtain the ranked rectifications list from the diagnostic controller, use the "rankedRects" message.